Análisis de Dempeño de un Modelo de Regresión Logística haciendo uso de Frameworks

Marcela Ibarra Mora A0123973

ABSTRACT

Las Regresiones logísticas son herramientas las cuales se utilizan para realizar modelos matemáticos los cuales sean capaces de realizar predicciones sobre variables categóricas. Estos modelos se entrenan con diferentes técnicas para obtener la menor cantidad de error al momento de realizar predicciones. Por otro lado, es importante identificar cuando el modelo se comporta de manera incorrecta para poder realizar los cambios necesarios a los diferentes hiperparametros y obtener resultados mucho más precisos.

I. INTRO

Dentro del aprendizaje supervisado existen dos ramas la cuales son las bases de muchos de los algoritmos que se utilizan actualmente dentro de Aprendizaje Máquina, esas dos bases son la Regresión Lineal y Regresión Logística. Específicamente la Regresión Logística es capaz de clasificar datos. Este tipo de algoritmos cuentan con hiperparametros los cuales son modificados durante el proceso de entrenamiento para poder predecir variables categóricas. Por lo tanto es importante también ser capaces de identificar los momentos en los que el modelo no se comporta de manera correcta. En este reporte se enfocarán en dos tipos de comportamientos incorrectos los cuales puede tener el modelo, estos siendo, el nivel de sesgo o bias y el nivel de varianza, para después poder identificar la clasificación del modelo, ya sea como underfitting, fitting y overfitting. Y finalmente aplicar alguna técnica de regularización o ajuste de parámetros para mejorar el desempeño del modelo

II. Data Set

El data set se compone de 178 datos sobre diferentes vinos con 14 columnas conformadas por diferentes variables las

cuales nos ayudarán a predecir la clasificación de 3 tipos de vino. Contamos con 3 clases: 1,2 y 3. Para este modelo se optó por no utilizar las 13 columnas o features, sino que se tomaron dos. Los features elegidos fueron Alcohol y Color intensity, las cuales nos indican la cantidad de alcohol del vino y la intensidad de color.

```
#Import libraries
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
```

```
columns = ["class","Alcohol","Malic aci
d","Ash","Alcalinity of ash","Magnesiu
m","Total phenols","Flavanoids",
"Nonflavanoid phenols","Proanthocyanin
s","Color intensity","Hue","OD280/OD315
 of diluted wines","Proline"]
df = pd.read_csv('wine.data',names=colum
ns)
#clean data
df_clean = df.drop(["Malic acid","As
h","Alcalinity of ash","Magnesium","Tota
l phenols","Flavanoids",
"Nonflavanoid phenols","Proanthocyanin
s","Hue","OD280/OD315 of diluted wine
s","Proline"], axis=1)
x = df_clean[["Alcohol","Color intensit
y"]].to_numpy()
y = df_clean["class"]
```

	class	Alcohol	Color intensity
0	1	14.23	5.64
1	1	13.20	4.38
2	1	13.16	5.68
3	1	14.37	7.80
4	1	13.24	4.32
...	...	...	...
173	3	13.71	7.70
174	3	13.40	7.30
175	3	13.27	10.20
176	3	13.17	9.30
177	3	14.13	9.20

178 rows × 3 columns

Figura 1. Tabla df_clean. Data set limpia.

III. Regresión Logística

La Regresión Logística es un algoritmo el cual es capaz de obtener la probabilidad de que la entrada pertenezca a cierta clase. Para cada una de las entradas (x) tenemos un parámetro que nos ayuda a predecir una salida binaria (y), así como una beta que identifica la intersección con el eje y.

$$y = m_1 x_1 + m_2 x_2 + b$$

Durante el proceso de entrenamiento lo que se hace es modificar estos parámetros y el bias para hacer predicciones, así como el uso de una función sigmoide la cual nos ayuda a

identificar a qué clase pertenece cada dato.

IV. Subgrupos de Entrenamiento, Prueba y Validación

Los datos se dividieron en 3 grupos: el subgrupo de Entrenamiento, el cual se utiliza para entrenar el modelo y así poder modificar los parámetros del modelo. El subgrupo de Prueba se utiliza para verificar si el modelo puede hacer predicciones y principalmente se utiliza para hacer una evaluación de cómo se comporta el modelo, este subgrupo se cuenta como que no está sesgado, ya que no es utilizado hasta que el modelo está completamente entrenado. Por otro lado tenemos el subgrupo de validación el cual también se utiliza para ver el comportamiento del modelo pero este se utiliza dentro del entrenamiento para poder modificar los parámetros o hiperparametros, este grupo si tiene cierto sesgo debido a que el modelo si tiene acceso a estos datos (puede “ver los datos”) por lo tanto tiene las respuestas a las predicciones y puede aprender de ellos.

```
#Sklearn function to split the data
from sklearn.model_selection import
train_test_split
```

```
#20% of the data is for testing and
#80% is for training
X_train, X_test, y_train, y_test = train
_test_split(x, y, test_size=0.2, random_
state=1)

#from the 80% of training data the (113
 instances for traing)
#20% is for validation
X_train, X_val, y_train, y_val = train_t
est_split(X_train, y_train, test_size=0.
25, random_state=1) # 0.25 x 0.8 = 0.2
```

Una vez separados los datos el modelo se puede entrenar.

```
#import logistic regresion model from sk
learn
from sklearn.linear_model import Logisti
cRegression
#Create the model, fit_intercept indicat
es that the model has the intercept
model = LogisticRegression(fit_intercept
=True)
#Train the model
model.fit(X_train,y_train)
#Test de model
y_model = model.predict(X_test)
#import the accuracy_score function to s
ee the behavior of the model
from sklearn.metrics import accuracy_sco
re
print(accuracy_score(y_test,y_model))
```

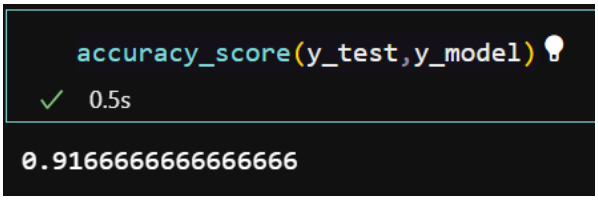

Figura 2. Accurracy score del modelo de regresión Logística

V. Grado de bias o sesgo y varianza

Como se puede visualizar en el código anterior, el accuracy o precisión del modelo es de 0.91 o 91% por lo tanto es capaz de hacer predicciones correctas. Por otro lado, esta acción puede indicarnos que el modelo es muy bueno prediciendo los datos de entrenamiento y prueba, pero eso no nos asegura que pueda predecir datos los cuales nunca haya visto. Por esa razón se deben de hacer diferentes análisis para determinar la eficiencia del modelo.

Primero se se realizó una matriz de confusión para determinar cuántas predicciones acertadas tuvo el modelo y cuantas equivocadas:

```
#import function for confusion Matrix
from sklearn.metrics import confusion_ma
trix
confusion_matrix(y_test, y_model)
```

```
array([[14,  0,  0],
       [ 0, 13,  0],
       [ 2,  1,  6]], dtype=int64)
```

Figura 3. Matriz de Confusión

Como se puede observar en la matriz, 33 de las 36 predicciones fueron correctas, la clase la cual el modelo tiene más dificultad de predecir es la 3.

El bias o el sesgo es la medida que nos indica el promedio de qué tan diferente es la predicción del modelo que tenemos de la predicción del mejor modelo de estimación. Por otro lado, la variabilidad mide la variabilidad del modelo cuando es entrenado con diferentes instancias del mismo problema.

Graficar la curva de aprendizaje nos ayuda a darnos cuenta si tenemos alta o baja varianza o bias:

```
#import library
from sklearn.model_selection import lear
ning_curve

train_sizes, train_scores, test_scores =
learning_curve(model, x, y, n_jobs=-1, t
rain_sizes=np.linspace(0.1, 1.0, 100))

train_mean = np.mean(train_scores, axis=
1)
train_std = np.std(train_scores, axis=1)

test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

#Plot the learning curve
plt.plot(train_sizes, train_mean, color
='blue', marker='o', markersize=5, label
='Training Accuracy')
plt.fill_between(train_sizes, train_mean
+ train_std, train_mean - train_std, alp
ha=0.15, color='blue')
```

```
plt.plot(train_sizes, test_mean, color
='green', marker='+', markersize=5, line
style='--', label='Validation Accuracy')
plt.fill_between(train_sizes, test_mean
 + test_std, test_mean - test_std, alpha
=0.15, color='green')
plt.title('Learning Curve')
plt.xlabel('Training Data Size')
plt.ylabel('Model accuracy')
plt.grid()
plt.legend(loc='lower right')
plt.show()
```

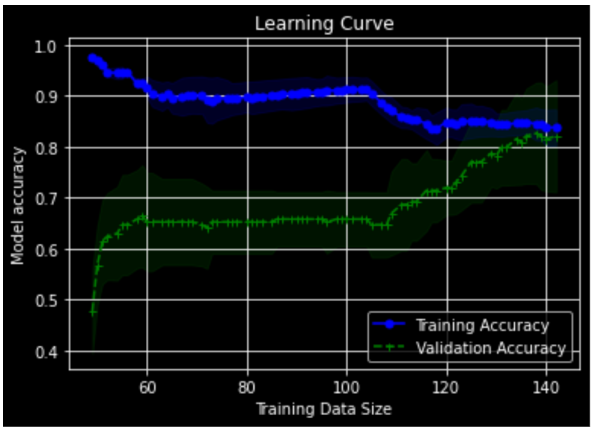

Figura 4. Curva de aprendizaje del modelo

Observando la gráfica se puede ver cómo cambia el accuracy del modelo al aumentar el tamaño de datos de entrenamiento. Se puede observar que cuando tenemos menos de 60 datos de entrenamiento, tenemos un comportamiento de overfitting alto debido a que el accuracy de los datos de entrenamiento baja de manera rápida y el accuracy de los datos de validación sube rapidamente, asi como que su diferencia en muy alta, lo cual nos indica un bias alto. Por otro lado, cuando tenemos más de 60 datos pero menos de 113 datos el modelo se estabiliza y empieza a tener un mejor comportamiento. Esto lo podemos ver cuando entrenamos los datos anteriormente, teníamos 113 datos para hacer el entrenamiento y el accuracy obtenido al hacer la prueba del modelo con los datos de prueba fue alrededor de 0.91. Así mismo, al entrenar el modelo con más de 113 datos el modelo vuelve al mismo comportamiento de overfitting.

V. Mejoras al modelo

Para resolver estos casos donde se tiene comportamiento donde tenemos overfitting, hay varias soluciones:

- Simplificar el modelo al quitar parámetros
- Modificar los hiper parámetros como el taza de aprendizaje
- Conseguir más datos
- Entre otros